Unificación de Modelos Evolutivos

Juan C. Vázquez
Universidad Tecnológica Nacional,
Facultad Regional Córdoba
Departamento de Ingeniería en Sistemas de Información
Córdoba, Argentina, 5016
jcvazquez@sistemas.frc.utn.edu.ar

Abstract

Evolutionary algorithmic mimics the way in which nature has solved problems through millions of years. The general method seems to be trial and error adaptation, generation after generation, resulting in survival of the fittest. This is called optimization process in engineering. Neural networks, genetic algorithms, cellular automata and other computational devices, use this strategy to attack discrete systems, estimated, essentially, as complex, non lineal and uncertain.

Our project attempts to clearly establish this essential relationship between different approaches, showing that they are different sides of the same coin. To think in neural networks as systems that evolve a cellular automata through learning, in cellular automata evolving as calculus from a differential equations system that obtain synaptic weights from a trained neural network. We attempt to determine a common phenotype in order to develop a genotype that explains and explicit it.

Keywords: Artificial Neural Network, cellular automata, evolutionary models.

Resumen

La algorítmica evolutiva imita la forma en que la naturaleza ha resuelto sus problemas durante millones de años. Su método general parece ser el de adaptación por prueba y error, generación tras generación, logrando el éxito del más apto. En ingeniería llamamos a esto un proceso de optimización. Tanto las redes neuronales, los algoritmos genéticos, los autómatas celulares y otros dispositivos computacionales, implementan esta estrategia para atacar sistemas discretos que se estiman complejos, no lineales e inciertos, esencialmente.

Nuestro estudio intenta establecer claramente esta relación esencial entre los distintos enfoques, mostrando que son caras distintas de una misma moneda. Pensar las redes neuronales como sistemas que durante su aprendizaje evolucionan algún autómata celular, los autómatas celulares evolucionando como cálculos de algún sistema de ecuaciones diferenciales que obtienen pesos sinápticos de una red neuronal entrenada. Intentamos determinar un fenotipo común para luego desarrollar un genotipo que lo explique y explicite.

Palabras claves: Redes neuronales artificiales, autómatas celulares, modelos evolutivos.

1. INTRODUCCIÓN

La algorítmica evolutiva imita la forma en que la naturaleza ha resuelto sus problemas durante millones de años. Su método general parece ser el de adaptación por prueba y error, generación tras generación, logrando el éxito del más apto. En ingeniería llamamos a esto un proceso de optimización.

Tanto científicos como ingenieros, intentan modelar los sistemas bajo estudio utilizando ecuaciones matemáticas (en general diferenciales) o dispositivos abstractos (máquinas, autómatas, grafos, etc.) para poder interrogar al modelo, en una primera instancia para validarlo al contrastar su funcionamiento con ejemplos conocidos de la realidad y luego, para simular la realidad y así comprenderla mejor y eventualmente controlarla.

Tanto las redes neuronales, los algoritmos genéticos, los autómatas celulares y otros dispositivos computacionales, implementan la estrategia de la naturaleza para atacar sistemas discretos en los que se estima que su complejidad, no linealidad e incertidumbre son esenciales, por lo que un modelo matemático no es factible de ser construido.

Nuestro estudio intenta establecer claramente esta relación esencial entre los distintos enfoques, mostrando que son caras distintas de una misma moneda. Pensar las redes neuronales como sistemas que durante su aprendizaje evolucionan algún autómata celular, los autómatas celulares evolucionando como cálculos de algún sistema de ecuaciones diferenciales que finalmente obtienen pesos sinápticos de una red neuronal entrenada. Intentamos determinar un fenotipo común para luego desarrollar un genotipo que lo explique y explicite.

En el proceso hemos aplicado algunas de nuestras herramientas, transfiriéndolas al modelado de problemas sociales de las características expuestas. Concretamente a la determinación del riesgo para la salud de la vivienda urbana, que bajo un enfoque holístico de la salud comunitaria, resulta ser un fenómeno que los expertos demógrafos consideran complejo y altamente no lineal por involucrar factores humanos, sociales y geopolíticos.

2. MODELOS BAJO ESTUDIO

2.1. Autómata Celular Unidimensional

En el caso más simple, un autómata celular consiste de un arreglo unidimensional de celdas contiguas que pueden contener cada una un valor entero entre **0** (cero) y ***k-1*** para $\mathbf{k} \in \mathbf{Z}^+$, $\mathbf{k} \geq \mathbf{2}$. En el caso **k = 2** llamamos al autómata celular ***elemental***.

t	**•••**	**0**	**0**	**1**	**0**	**0**	**•••**

Figura 1: Autómata celular unidimensional elemental

Los valores de las celdas evolucionan sincrónicamente en etapas de tiempo discreto, de acuerdo a una regla sencilla idéntica para todas las celdas del autómata (usualmente una función booleana en autómatas celulares elementales); ésta involucra solamente el valor contenido en la celda misma y en sus ***r*** vecinas inmediatas a ambos lados, en el instante anterior; ***r*** recibe el nombre de ***rango de la regla*** y hablamos en este caso de una **r-vecindad**.

t + 1	**•••**	**0**	**1**	**0**	**1**	**0**	**•••**

$$a_i^{(t+1)} = a_{i-1}^{(t)} \vee a_{i+1}^{(t)}$$

Figura 2: Autómata de la figura 1, evolucionado en un paso bajo la suma lógica

En la figura 2, el autómata celular es elemental (***k = 2***, ***r = 1***), su regla de evolución es la suma lógica de los valores de la celda inmediata anterior e inmediata posterior a la celda en proceso y el rango de la regla es **1**. El subíndice de la notación corresponde a la posición de la celda dentro del arreglo y el superíndice entre paréntesis indica la etapa del valor accedido. Nótese que para celdas binarias (sólo valores cero o uno) y una 1-vecindad (tres celdas), pueden establecerse 256 reglas distintas.

El arreglo se supone infinito, por lo que a los fines prácticos debe elegirse un número entero positivo ***N*** de celdas e imponerse una condición de contorno a las celdas extremas del arreglo finito, la cual puede consistir en mantenerlas en un valor fijo (activo o no-activo) o en tomar el arreglo como circular.

Desde el punto de vista de la Teoría de la Computación, podemos definir el AC como una máquina abstracta. Así, formalmente definimos el caso general de un autómata celular unidimensional como la quíntupla:

$$(\Sigma, N, C, r, \varphi)$$

donde:

- Σ es un conjunto finito y no vacío de ***k*** símbolos, denominado **alfabeto** del autómata. Usualmente está compuesto por los enteros de **0** a **k-1**.
- **N** es la cantidad de celdas del autómata celular.
- **C** especifica la condición de contorno impuesta a las celdas extremas. Normalmente se mantiene su valor fijo igual al neutro de la función φ o se establece circularidad asumiendo que a la última celda le sigue la primera y viceversa.
- **r** es el tamaño de la vecindad que incide sobre el próximo valor de las celdas.
- $\varphi:\Sigma^{2r+1}\rightarrow\Sigma$ es la regla de actualización de los valores de las celdas que tomando **r** celdas a cada lado de la celda objetivo y ella misma, calcula su nuevo valor.

No se ha incluido un conjunto de estados, común en las máquinas abstractas; en realidad, el autómata sí puede decirse que tiene un estado en cada instante de tiempo constituido por las cadenas $\alpha_t \in \Sigma^N$ de símbolos del alfabeto que se encuentran en el instante **t** en las celdas; normalmente esto es referido en la bibliografía como la ***configuración*** del autómata y no como su estado. La aplicación de φ a cada celda del autómata en el tiempo **t** traduce la cadena α_{t-1} en la cadena α_t por lo que podríamos hablar inclusive de una función $\Phi:\Sigma^N \rightarrow \Sigma^N$ de transición de estado a estado, pero creemos que la definición dada es más clara en este punto de nuestro estudio.
Dada una configuración inicial α_0 arbitraria, a medida que transcurre el tiempo (discreto), se genera una sucesión de configuraciones α_0, α_1, … α_t, … que denominamos ***evolución espacio-temporal*** del autómata celular.
Una de las características de estas máquinas que presentan interés, es el tipo de evolución espacio-temporal que desarrollan; a partir de una configuración inicial aleatoria se generan estructuras que conforman ciertos patrones, contradiciendo la segunda ley de la termodinámica, ya que disminuye su entropía con el tiempo; si bien esto no es un sistema físico esta aparente contradicción se debe a que la evolución del AC es un fenómeno irreversible. Wolfram [Wolfram-1984] catalogó estas estructuras en **cuatro categorías** y especuló sobre su relación con los cuatro tipos de lenguajes formales en la jerarquía de Chomsky (regulares, independientes del contexto, sensibles al contexto, con estructura de frase) [Hopcroft-1979] y con los cuatro tipos de máquinas de estado finito que reconocen esos mismos lenguajes (autómatas finitos, autómatas con pila, autómatas linealmente acotados y máquinas de Turing).

Fig 3. Autómata tipo II de Wolfram

Para nuestro trabajo, es de particular interés el ***TIPO II DE WOLFRAM*** que es una clase de autómatas celulares que evolucionan con patrones irregulares o no definidos al principio, pero que a partir de un instante y en adelante llegan a una configuración fija y estable, como el que se muestra en la figura. Se verá más adelante porqué esto es importante para nosotros.

2.2. Redes Neuronales Artificiales

En Inteligencia Artificial, se proponen básicamente dos enfoques para lograr de un programa comportamiento inteligente: el ***enfoque simbólico*** sostiene que el objetivo se logrará emulando la forma en que (se cree que) los humanos piensan, esto es, haciendo manipulación de símbolos mediante reglas lógicas y procesando con esta metodología grandes bases de conocimiento; por otro lado, el ***enfoque subsimbólico*** propone emular por hardware o software la forma en la que la naturaleza resuelve los problemas, en particular, el llamado modelo conexionista que emula el "hardware" del sistema nervioso, con la esperanza de que el comportamiento inteligente surgirá como una propiedad *emergente* de la complejidad de millones de elementos simples cooperando interconectados.

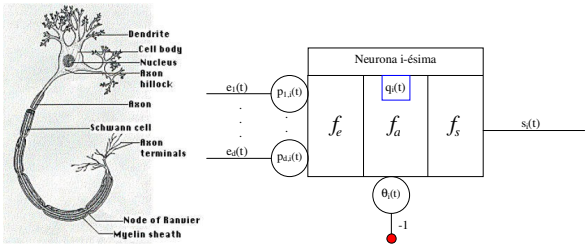

Fig. 4: Neurona biológica y modelo de neurona artificial

Una red neuronal artificial es un modelo computacional inspirado en el sistema nervioso de los animales superiores (en realidad en una simplificación extrema del mismo), respondiendo al enfoque subsimbólico de la IA.
Consiste de una cantidad importante de elementos de modesto poder de cálculo (las neuronas) fuertemente interconectadas por canales que permiten el transporte de información entre ellas (axones y dendritas).
Cada neurona recibe entradas ***E(t)*** y las procesa aplicándoles funciones de entrada, activación y salida (f_e, f_a, f_s respectivamente en la figura 4) en secuencia, obteniendo una salida ***S(t)*** en cada instante ***t***.

Se han diseñado infinidad de topologías distintas de conexionado de redes neuronales y aún muchos tipos distintos de neuronas. En [Hilera-1995], [Brío-1997] y otros autores que figuran en la bibliografía puede encontrarse una extensa y variada clasificación de redes neuronales.
Para determinar una red neuronal artificial se debe:

- Definir un conjunto finito y no vacío de elementos de cálculo (neuronas).
- Establecer la topología de la red (cómo interactuarán las neuronas entre ellas).
- Dar la dinámica de actualización de la red (sincrónica o asincrónica).
- Elegir una tarea a realizar por la red (pares entrada/salida de ejemplo).
- Generar un algoritmo para que la red aprenda la tarea (aprendizaje).

Nuestro estudio específicamente ha utilizado en su primer fase, redes perceptron multicapa (fig. 5) dejando para la segunda fase, el tratamiento de redes autoasociativas de tipo Hopfield.

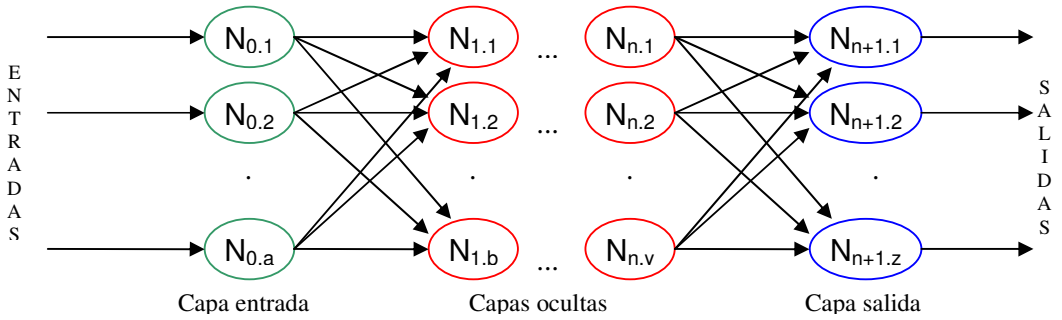

Fig. 5: Red perceptron multicapa

En estas redes, las neuronas están organizadas en capas y la comunicación es siempre hacia delante (feedforward); la primera capa que recibe las entradas desde el exterior, se denomina *capa de entrada*, luego puede haber cero o más capas de *neuronas ocultas* y la última capa, que comunica con el exterior, se denomina *capa de salida*.
Cada neurona de la figura 5, responde al modelo indicado en la figura 4, siendo los parámetros $p_{i,j}$ los pesos sinápticos que miden la fuerza de conexión entre pares de neuronas.
La red, antes de ser productiva, deben pasar por una *etapa de aprendizaje* que, de alguna forma, calcule *el peso de cada conexión sináptica* y los *umbrales de cada neurona* de la red, de tal manera que la red ***infiera*** la relación existente entre la entrada y la salida de los diferentes juegos de entrenamiento.
El proceso de aprendizaje en una red de este tipo se realiza mediante un proceso iterativo denominado ***retro-propagación de errores***. Matemáticamente, este algoritmo recurre a la técnica de ***descenso por el gradiente***, que intenta disminuir en cada iteración el error producido por la red, medido como la diferencia entre la salida esperada y la salida calculada por la red; la función de error total se toma como el error medio cuadrático que produce la red y es una función sólo de los pesos sinápticos, supuesta la topología de conexionado y la cantidad y tipo de neuronas como parámetros fijos.

3. MOTIVACIÓN Y OBJETIVOS

El algoritmo de retropropagación de errores realiza su trabajo cambiando en cada iteración el valor de los pesos sinápticos para que el error total de la red disminuya paulatinamente, logrando que las salidas reales calculadas desde las entradas ejemplo, se acerquen tanto como se quiera a las salidas conocidas para esas entradas. Nuestra suposición es que, si la red aprende su tarea, en algún momento estos pesos deben dejar de variar y mantenerse fijos o a lo sumo variando en forma oscilante (con una pequeña oscilación) alrededor de un valor fijo o asintóticamente hacia él; si esto ocurre debería poder mostrarse esto como la evolución de un autómata celular del **Tipo II de Wolfram** y si esto es posible, tenemos la relación que estamos buscando: ***puede verse el proceso de aprendizaje de la RNA como un proceso evolutivo, la evolución de un AC***.

3.1. Objetivo General.

Determinar posibles relaciones entre los patrones emergentes de los AC y las RNA.

3.2. Objetivos Particulares.

Para lograr nuestro objetivo general, diseñamos una serie de pasos a seguir con propósitos definidos y acotados; a continuación enumeramos algunos de ellos:

- Lograr codificaciones significativas de la evolución del conocimiento obtenido por redes de tipo *backpropagation* durante su aprendizaje (primera fase) y de la evolución de los estados de activación de neuronas hacia el reconocimiento de patrones almacenados en redes tipo *Hopfield* (segunda fase).
- Establecer una representación gráfica uniforme de patrones evolutivos, que permita la clara comparación entre los mismos.
- Construir programas de computación que implementen estos modelos de redes neuronales y apliquen las codificaciones establecidas, mostrando gráficamente su evolución.
- Construir programas de computación que implementen autómatas celulares parametrizados para el uso de distintas reglas y tipos de vecindades, mostrando gráficamente su evolución.
- Efectuar la búsqueda de semejanzas entre los patrones emergentes de los AC y las RNA.
- De encontrarlas, determinar posibles relaciones matemáticas que expliquen estas semejazas.
- Aproximar un modelo formal de las relaciones encontradas y probar el modelo teórico.

4. METODOLOGÍA

Siguiendo la línea de Wolfram (Wolfram-1994) que modela matemáticamente, visualiza patrones generados por los modelos traducidos a programas de computadoras y descubre similitudes de comportamiento analizándolos, se utilizó la experimentación computacional, esto es:

- Se efectuó un desarrollo formal detallado de estos modelos desde la perspectiva de la teoría de autómatas y el estudio de sus propiedades matemáticas y computacionales, para determinar qué variables serán controladas durante el estudio [Hopcroft-1979] [Wolfram-1994].
- Se generaron distintas codificaciones para representar la evolución del conocimiento que redes de tipo BP adquieren durante su fase de aprendizaje (asignación de seis funciones de conversión de los pesos sinápticos, secuenciación de neuronas según orden de capas, etc.).
- Se construyeron prototipos operativos de redes BP y HP para reconocimiento de símbolos (casos simples) y de AC para muestreo.
- Se realizaron pruebas de evolución de variables bajo estudio (pesos sinápticos, estado de neuronas, estado de células, etc.) y estudio de las mismas (límites de valores, periodicidad, estabilidad, etc.).
- Se determinó la representación gráfica a utilizar (granularidad, colores, intervalos de muestreo y graficación, precisión de valores generados, etc.) de tal manera que sea uniforme y significativa en todos los modelos bajo estudio.
- Se diseño e implementó un algoritmo para cotejo computacional de resultados.
- Se experimentó con los programas construidos.

5. RESULTADOS

Los experimentos realizados consistieron en la generación de archivos planos con los valores que adoptaban los pesos sinápticos durante el aprendizaje en redes MLP, su posterior conversión a cadenas binarias y análisis de aproximación a la evolución espacio-temporal de autómatas celulares mediante el programa de descubrimiento.

En todos los casos hasta ahora, las únicas veces que se logró la concordancia buscada fue con archivos codificados según el tercer esquema de codificación, esto es, tomar el peso sináptico que es un número real entre 0 y 1, multiplicarlo por la unidad seguida de ceros para transformar las cifras más significativas a entero y luego convertir ese entero en binario puro según la representación larga de enteros de C#. Con las otras codificaciones nunca obtuvimos concordancia.

El cuadro de la figura 6 muestra algunos de los resultados arrojados por el programa descubridor en aquellas oportunidades que se tuvo éxito. Se asumió un porcentaje de concordancia de más del 90% como éxito en la búsqueda de un patrón.

```
# 2a8_3221
# RNA-AC: BackPropagation
0               1 = 5.875952121871597 %       0 = 94.12404787812841 %
1               1 = 93.68863955119214 %       0 = 6.311360448807854 %

# 2a8_3241
# RNA-AC: BackPropagation
0               1 = 8.481927710843372 %       0 = 91.51807228915662 %
1               1 = 91.06493506493507 %       0 = 8.935064935064934 %

# 2a8_3241
# RNA-AC: BackPropagation
0               1 = 5.899198167239406 %       0 = 94.1008018327606 %
1               1 = 92.2283356258597 %              0 = 7.771664374140302 %

# Dig_15_15_20_5_1
# RNA-AC: BackPropagation
0               1 = 5.408637873754152 %       0 = 94.59136212624585 %
1               1 = 94.09396632137071 %       0 = 5.9060336786292895 %
```

Figura 6: Resultados en corridas exitosas del descubridor de AC

En el cuadro de la figura 7, se muestra un ejemplo de concordancia al 100% parcialmente ya que la vecindad es grande y conlleva una regla muy complicada, pero existe !!!

```
# XOR_2241
# RNA-AC: BackPropagation
000000000000010               1 = 0.0 %      0 = 100.0 %
000000000000101               1 = 0.0 %      0 = 100.0 %
000000000001010               1 = 0.0 %      0 = 100.0 %
000000000010101               1 = 0.0 %      0 = 100.0 %
000000000100000               1 = 0.0 %      0 = 100.0 %
000000000101011               1 = 0.0 %      0 = 100.0 %
000000001000000               1 = 0.0 %      0 = 100.0 %
000000001010110               1 = 0.0 %      0 = 100.0 %
000000010000001               1 = 100.0 %    0 = 0.0 %
000000010001110               1 = 100.0 %    0 = 0.0 %
000000010100001               1 = 100.0 %    0 = 0.0 %
000000010101100               1 = 100.0 %    0 = 0.0 %
000000011001000               1 = 0.0 %      0 = 100.0 %
000000011010000               1 = 100.0 %    0 = 0.0 %
000000100000011               1 = 0.0 %      0 = 100.0 %
000000100001110               1 = 0.0 %      0 = 100.0 %
000000100011101               1 = 0.0 %      0 = 100.0 %
000000101000011               1 = 0.0 %      0 = 100.0 %
000000101011000               1 = 0.0 %      0 = 100.0 %
000000110000010               1 = 100.0 %    0 = 0.0 %
000000110010000               1 = 100.0 %    0 = 0.0 %
000000110010001               1 = 0.0 %      0 = 100.0 %
000000110010010               1 = 100.0 %    0 = 0.0 %
000000110100000               1 = 100.0 %    0 = 0.0 %
000001000000110               1 = 0.0 %      0 = 100.0 %
000001000010100               1 = 0.0 %      0 = 100.0 %
000001000011100               1 = 0.0 %      0 = 100.0 %
000001000111010               1 = 0.0 %      0 = 100.0 %
000001001101100               1 = 0.0 %      0 = 100.0 %
000001010000111               1 = 100.0 %    0 = 0.0 %
000001010110000               1 = 100.0 %    0 = 0.0 %
000001100000100               1 = 0.0 %      0 = 100.0 %
000001100100000               1 = 0.0 %      0 = 100.0 %
000001100100001               1 = 100.0 %    0 = 0.0 %
000001100100011               1 = 0.0 %      0 = 100.0 %
000001100100100               1 = 0.0 %      0 = 100.0 %
```

… y siguen muchísimas más líneas.

Figura 7: Parte de la regla del AC deducida desde la evolución de pesos sinápticos

Estas reglas complicadas siempre se encontraron en archivos representativos del problema XOR codificados según el esquema 3.

6. CONCLUSIONES

Hemos experimentado bastante con las herramientas generadas, pero no hemos llegado aún a la sistematización completa de los resultados obtenidos en estos experimentos como para generar una razonable casuística; los resultados preliminares obtenidos hasta la fecha son alentadores ya que en varios casos hemos encontrado una clara relación entre las RNA y los AC, pero debemos seguir experimentando y documentando resultados.
Hemos detectado que con el tercer método de codificación existe una relación en un porcentaje aceptable y a veces exacto, aunque la misma no se ha conseguido con el archivo completo de pesos sinápticos codificados en binario sino con el último diez por ciento de los registros anteriores a la estabilización de pesos.
Hay fuertes problemas inherentes al manejo de números reales en los lenguajes de programación donde las últimas cifras varían a veces de formas imprevistas quedando sólo las primeras como válidas y a veces ni éstas. Un número como 0,9 puede ser transformado por el lenguaje en 0,89999999999999 al ser almacenado en una variable double; esto numéricamente es casi lo mismo pero cuando estos números se convierten a cadenas de ceros y unos y se los compara, el impacto es altísimo.
Por lo expuesto, **no podemos asegurar a la fecha** con nuestras codificaciones de números reales en cadenas binarias y nuestros programas de experimentación que exista la relación que queríamos encontrar en forma certera, pero creemos que aún podemos lograrlo por lo cual seguiremos experimentando y revisando nuestros resultados.

Nos quedan varios puntos por pulir y algunas ideas para seguir investigando esta temática a saber:

- Hay que revisar todos los programas e incluir técnicas que minimicen el problema de manejo de decimales significativos ante expuesto.
- Se debe mejorar el programa de descubrimiento para que brinde mayor información al detectar posible concordancia (cantidad y rango de registros realmente tomados para el cálculo de porcentaje, por ejemplo).
- Hay que seguir experimentando y registrando resultados para lograr una estadística razonablemente válida.

REFERENCIAS

[Avnet-2000] Avnet J. (2000),
Computation, Dynamics and the Phase-Transition, http://www.theory.org/complexity/cdpt,
[Fecha de consulta : 17/07/2003]

[Castro-2002] Castro J., Mantas C., Benítez J. (2002),
Interpretation of Artificial Neural Networks by Means of Fuzzy Rules,
IEEE Transactions on Neural Network, Vol. 13, #1, pp. 101-116.

[Darwin1859] Darwin, C. (1859),
On the origins of species by means of natural selection, or the preservation of favoured race in the struggle for life, Traducción: *El origen de las especies,* (1992) Editorial Planeta, Barcelona, España.

[Das-1995] Das R., Crutchfield J., Mitchel M., Hanson J. (1995),
Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kaufmann, pp. 336-343.

[Denker-1987] Denker J., Shwartz D., Wittner B., Solta S., Howard R., Jackel K., Hopfield J. (1987),
Large Automatic Learning, Rule Extraction and Generalization, Complex Systems, Vol. 1, pp. 877-922.

[Deolikar-2002] Deolalikar V. (2002),
A Two-Layer Paradigm Capable of Forming Arbitrary Decision Regions in Input Space,
IEEE Transactions on Neural Network, Vol. 13, #1, pp. 15-21.

[Hopcroft-1979] Hopcroft J. E.y Ullman J. D. (1979),
Introduction to Automata Theory, Language and Computation, Addison-Wesley, New York, USA.

[Newman-1994] Newman J. (1994),
El Mundo de las Matemáticas, Vol. 6, Grijalbo, Barcelona, España.

[Rumelhart-1986] Rumelhart D. y McClelland J. (1986),
Parallel Distributed Processing, Vol. 1: *Foundations*, MIT Press.

[Wolfram-1984] Wolfram S. (1984),
Universality and Complexity in Celullar Automata, Physica D, vol. 10, pp. 1-35.

[Wolfram-1985] Wolfram S. (1985),
Twenty Problems in Theory of Cellular Automata, Physica Scripta, vol. T9, pp. 170-183.

[Wolfram-1994] Wolfram S. (1994),
Cellular Automata and Complexity (collected papers), Addison-Wesley, Massachusetts, USA.